1Б

УДК 681.324

А.А. Баркалов[1], С.А. Ковалев[2], Р.М. Бабаков[3], Д.В. Николаенко[2]
[1] Институт компьютерной инженерии и электроники, г. Зеленая Гура, Польша
[2] Донецкий национальный технический университет, Украина
[3] Государственный университет информатики и искусственного интеллекта, г. Донецк, Украина
a.barkalov@iie.uz.zgora.pl
cpld@mail.ru

Эвристический алгоритм оптимизации размещения микрокоманд в композиционном микропрограммном устройстве управления с разделением кодов и кэш-памятью

Разработан эвристический алгоритм повышения эффективности использования модуля кэш-памяти в композиционном микропрограммном устройстве управления с разделением кодов, основанный на специальной адресации операторных линейных цепей. Предложен ряд стратегий объединения нескольких операторных цепей в одном блоке памяти, позволяющий в общем случае увеличить значение вероятности кэш-попаданий для граф-схемы реализуемого алгоритма управления. Рассмотрен пример использования предложенного эвристического алгоритма.

Общая постановка проблемы

Одним из структурных элементов современных вычислительных систем является устройство управления (УУ), которое может быть реализовано в виде композиционного микропрограммного устройства управления (КМУУ) с разделением кодов, в котором достигается минимальное число выходов схемы адресации [1]. Использование элементного базиса ПЗУ при реализации управляющей памяти (УП) удешевляет схему устройства, однако, в то же время приводит к значительному снижению быстродействия схемы [2]. Таким образом, для промышленности средств вычислительной техники актуальной научно-технической задачей является задача увеличения быстродействия схемы КМУУ. Решение данной задачи повлечет за собой увеличение быстродействия вычислительных систем, реализованных на базе КМУУ, и, как следствие, расширит область их применения.

Для увеличения быстродействия схемы КМУУ с разделением кодов в работе [3] предложен метод, заключающийся в использовании дополнительного модуля кэш-памяти микрокоманд для хранения наиболее часто используемых строк управляющей памяти. Использование кэш-памяти позволяет снизить среднее время доступа к управляющей памяти, что приводит к уменьшению средней длительности такта работы устройства и к увеличению его среднего быстродействия. При этом основным параметром, влияющим на эффективность использования кэш-памяти микрокоманд, является вероятность кэш-попаданий p_h, характеризующая отношение количества тактов, в которых произошло кэш-попадание, к общему количеству тактов работы устройства.

Характерной особенностью всех структур КМУУ является их аппаратная привязка к реализуемому алгоритму управления. Данная особенность позволяет провести оптимизацию схемы КМУУ для каждого конкретного случая реализации. В структуре КМУУ с разделением кодов и кэш-памятью, предложенной в [3], одним из факторов, влияющих на величину вероятности кэш-попаданий, является реализуемый алгоритм управления, традиционно представляемый в виде граф-схемы алгоритма (ГСА). Количество микрокоманд (МК), операторных линейных цепей (ОЛЦ) и переходов между ОЛЦ, а также сама структура ГСА – все это оказывает влияние на величину p_h. При этом на сегодняшний день неисследованными остаются факторы, влияющие на увеличение значения p_h и, как следствие, на увеличение среднего быстродействия устройства управления.

Эвристический подход к оптимизации размещения микрокоманд в управляющей памяти

В структурах КМУУ с разделением кодов имеют место естественная адресация микрокоманд внутри каждой ОЛЦ, а также тот факт, что в первой микрокоманде ОЛЦ адресные разряды, следующие после кода ОЛЦ, равны нулям [2]. По этим причинам процесс адресации микрокоманд, являющийся одним из этапов синтеза КМУУ, в структурах с разделением кодов сводится по сути к адресации ОЛЦ.

Пусть в заданной ГСА существует переход из ОЛЦ α_i в ОЛЦ α_j. При этом если обе ОЛЦ в данный момент находятся в кэш-памяти, то возникнет ситуация кэш-попадания, в противном случае – ситуация кэш-промаха. В том случае, если обе ОЛЦ расположены в управляющей памяти последовательно и принадлежат одному блоку памяти, они окажутся вместе в одной строке кэш-памяти, и упомянутый переход из α_i в α_j всегда будет приводить к кэш-попаданию. Если же данные ОЛЦ находятся в различных блоках памяти, то при выполнении перехода $\alpha_i \rightarrow \alpha_j$ возможна ситуация кэш-промаха, и значение p_h для ГСА будет несколько ниже. Если же одновременное нахождение данных ОЛЦ в кэш-памяти невозможно (например, при использовании кэш-памяти с прямым отображением данных), то данный переход всегда будет приводить к кэш-промаху, что еще более снизит величину вероятности кэш-попаданий для заданной ГСА.

Согласно принципу пространственной локальности данных, в случае кэш-промаха в кэш из УП помещается не только запрашиваемая микрокоманда, но и несколько микрокоманд из ближайших к ней адресов памяти. При этом количество загружаемых команд определяется размером строки кэш-памяти, а для кодирования микрокоманд внутри строки используется часть младших разрядов адреса микрокоманды. С учетом этого содержимое управляющей памяти может быть представлено в виде *блоков микрокоманд*, имеющих размер, равный размеру строки кэш-памяти и располагающихся последовательно, начиная с нулевого адреса. Деление УП на блоки является постоянным и не зависит от размещения микрокоманд в адресном пространстве УП.

Экспериментальным путем авторами выработаны четыре основных эвристических правила (эвристики), характеризующие пути повышения эффективности размещения ОЛЦ в блоках управляющей памяти:

Эвристика 1. Вероятность кэш-попаданий не зависит от того, какому по порядку блоку памяти принадлежит ОЛЦ. Иными словами, конкретные значения адресов памяти, занимаемые ОЛЦ, не оказывают влияния на вероятность кэш-попаданий. Из рассмотренного правила вытекает следующий вывод: если содержимое

блока памяти сместить в адресном пространстве УП вперед или назад на количество ячеек, кратное размеру блока, то значение вероятности кэш-попаданий не изменится.

Эвристика 2. Вероятность кэш-попаданий не зависит от порядка следования ОЛЦ в блоке. Эвристика действует и в том случае, если блок памяти содержит более двух ОЛЦ. Вывод, следующий из данного правила, может быть сформулирован следующим образом: если несколько ОЛЦ могут быть размещены в одном блоке УП, то порядок помещения ОЛЦ в блок не влияет на результирующее значение вероятности кэш-попаданий.

Для пояснения третьего эвристического правила введем следующие определения:

1. «Опасной» МК считается такая МК, при выполнении которой кэш-промах возможен (но не обязателен). МК, при выполнении которой кэш-промах невозможен ни при каких обстоятельствах, считается «безопасной». К «опасным» МК относятся микрокоманды, являющиеся входами ОЛЦ.

2. Вероятностью выполнения (или весом) $p(a_i)$ микрокоманды a_i будем называть отношение среднего количества выполнений $K(a_i)$ МК a_i за один проход алгоритма к среднему количеству микрокоманд K, выполняющихся за один проход алгоритма

$$p(a_i) = K(a_i) / K. \qquad (1)$$

3. «Опасный» вес $v(a_i)$ микрокоманды a_i равен сумме весов микрокоманд других ОЛЦ, из которых есть непосредственный переход в микрокоманду a_i, умноженных на вероятность данного перехода.

4. «Опасный» вес блока памяти равен суммарному «опасному» весу ОЛЦ, входящих в блок.

Эвристика 3. Если ОЛЦ O_i имеет переходы в ОЛЦ O_j, то размещение этих ОЛЦ в одном блоке памяти снижает «опасный» вес ОЛЦ O_j. Из эвристики 3 может быть сделан следующий вывод: наиболее целесообразно добавить в текущий блок ту ОЛЦ, при которой «опасный» вес текущего блока будет минимальным.

Эвристика 4. Увеличение количества ОЛЦ, на которые разбито множество микрокоманд рассматриваемой ГСА, не влияет на вероятности выполнения микрокоманд, но увеличивает количество вариантов размещения ОЛЦ в управляющей памяти.

Разработка эвристического алгоритма оптимизированного размещения ОЛЦ в управляющей памяти

Основываясь на полученных эвристиках, построим эвристический алгоритм оптимизации размещения ОЛЦ в адресном пространстве управляющей памяти с целью повышения значения вероятности кэш-попаданий.

Исходными данными при этом выступают:

- исходная ГСА;
- вероятности выполнения логических условий;
- размер строки данных в модуле кэш-памяти;
- количество и содержимое ОЛЦ.

Алгоритм имеет следующие основные этапы:

1. Определение вероятностей выполнения микрокоманд.
2. Отнесение всех ОЛЦ ко множеству нераспределенных ОЛЦ.

3. Считать текущим начальный блок УП.

4. Если текущий блок полностью заполнен, считать текущим следующий пустой блок.

5. Если текущий блок пустой, поместить в него одну из нераспределенных ОЛЦ.

6. Если текущий блок заполнен частично, добавить в него ту нераспределенную ОЛЦ, вместе с которой суммарный «опасный» вес блока будет минимальным.

7. Если остались нераспределенные ОЛЦ, перейти к пункту 4.

8. Конец.

Отметим следующее.

В пункте 5 алгоритма в пустой блок добавляется одна из нераспределенных ОЛЦ. Выбор нераспределенной ОЛЦ в каком-то смысле может быть произвольным: какая бы ОЛЦ ни была выбрана первой в блоке, алгоритм в силу эвристики 3 должен заполнить блок оптимальным способом. При этом те ОЛЦ, которые будут добавлены к блоку, обречены на «соседство» с первой выбранной ОЛЦ и варианты их «соседства» с другими ОЛЦ не рассматриваются. Таким образом, способ выбора первой ОЛЦ в блоке влияет на конечное содержимое блока, то есть на «опасный» вес блока и в конечном итоге на вероятность кэш-попаданий ГСА.

В настоящей работе предлагаются шесть вариантов способа выбора первой ОЛЦ для пустого блока, которые условимся называть стратегиями выбора нераспределенных ОЛЦ:

Стратегия 1. Выбирается ОЛЦ с максимальным весом.

Стратегия 2. Выбирается ОЛЦ с минимальным весом.

Стратегия 3. Выбирается ОЛЦ с максимальным «опасным» весом.

Стратегия 4. Выбирается ОЛЦ с минимальным «опасным» весом.

Стратегия 5. Выбирается ОЛЦ с максимальным отношением «опасного» веса к обычному весу.

Стратегия 6. Выбирается ОЛЦ с минимальным отношением «опасного» веса к обычному весу.

При использовании каждой из шести стратегий в общем случае может быть получен различный результат (вариант размещения микрокоманд и значение p_h). Безусловно, помимо предложенных, может быть найдено множество других стратегий, оказывающих различное влияние на результат алгоритма: например, стратегия, при которой первой всегда выбирается ОЛЦ, содержащая стартовую МК a_1, или стратегия случайного выбора. Поиск наиболее оптимальной стратегии выбора ОЛЦ является отдельно ветвью исследований и в настоящей работе не рассматривается.

Эвристический характер предложенного алгоритма заключается в следующем:

1. Оптимизация размещения производится для каждого блока в отдельности без учета связей между блоками. Большое количество переходов между ОЛЦ, находящимися в разных блоках, может приводить к увеличению количества кэш-промахов и снижению величины p_h.

2. Не учитывается количество строк в модуле кэш-памяти. Если кэш-память содержит лишь одну строку фиксированной длины, алгоритм замещения данных, предназначенный для выбора одной из нескольких строк кэш-памяти, не используется. В этом случае значение вероятности кэш-попаданий будет одинаковым для любой архитектуры кэш-памяти. Если же кэш-память содержит несколько строк, то влияние архитектуры кэш-памяти и алгоритма замещения данных на значение вероятности кэш-попаданий может быть значительным.

3. Не используется оптимальная стратегия выбора нераспределенной ОЛЦ. Хотя экспериментальные исследования показывают, что предложенные стратегии оказываются достаточно эффективными, нельзя говорить об абсолютной оптимальности какой-либо из них.

Пример использования эвристического алгоритма

Рассмотрим подробно пример оптимизации размещения содержимого управляющей памяти с использованием разработанного эвристического алгоритма и третьей стратегии выбора нераспределенной ОЛЦ.

Пусть алгоритм управления задан ГСА G (рис. 1). Пусть также известны вероятности переходов логических условий: $p(x_1) = 0{,}2$; $p(x_2) = 0{,}7$; $p(x_3) = 0{,}5$; $p(x_4) = 0{,}1$; $p(x_5) = 0{,}9$. Размер строки кэш-памяти будем считать равным 8 слов.

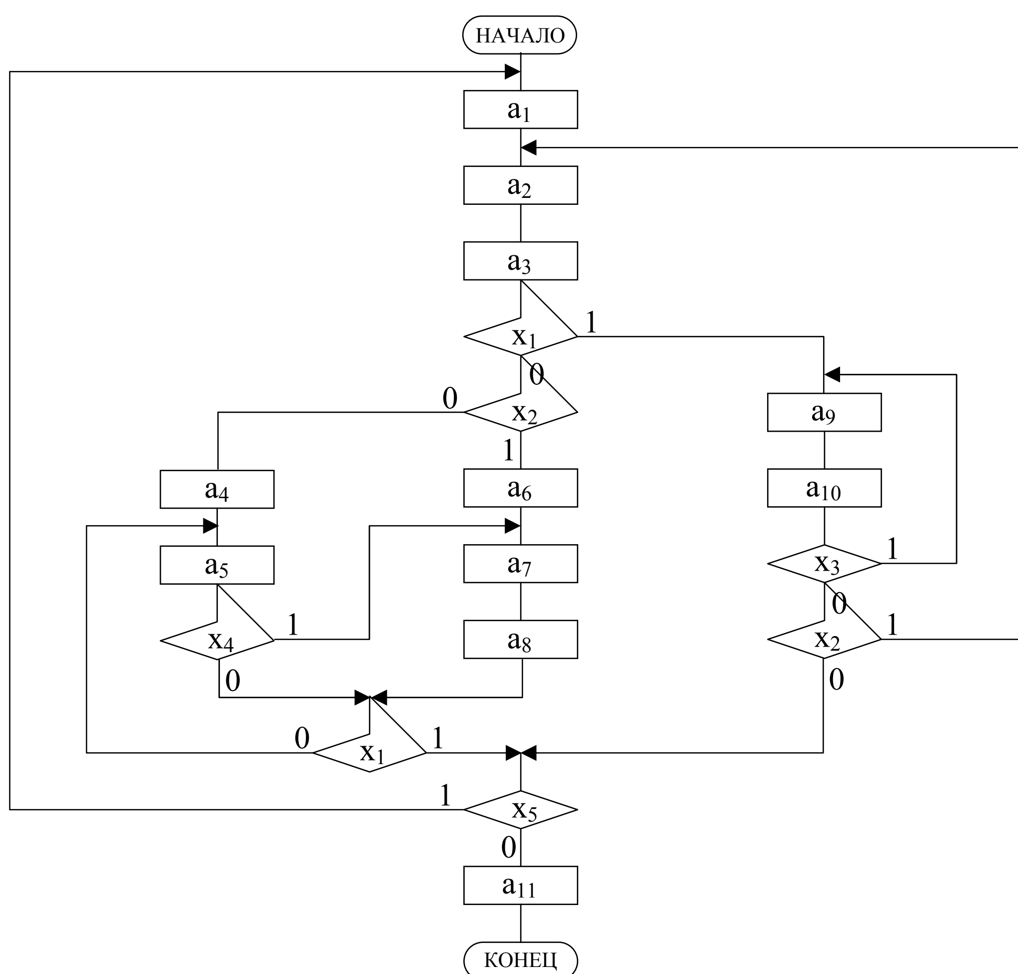


Рисунок 1 – Граф-схема алгоритма G

Сформируем ОЛЦ известным способом [2]: $O_1=\{a_1, a_2, a_3\}$, $O_2=\{a_4, a_5\}$, $O_3=\{a_6, a_7, a_8\}$, $O_4=\{a_9, a_{10}\}$, $O_5=\{a_{11}\}$. Максимальный размер ОЛЦ $N_{max} = 3$, следовательно, $R(T) =]\log_2(N_{max})[= 2$. Для пяти ОЛЦ $R(\tau) =]\log_2 5[= 3$. Разрядность адреса микрокоманды $R = R(T) + R(\tau) = 5$, емкость ПЗУ схемы УП составит $2^R = 32$ слова. При размере строки кэша $N_C = 8$ слов адресное пространство УП делится на четыре блока $B_1 – B_4$.

Выполняемые шаги эвристического алгоритма условимся обозначать в форме «**Шаг i, пункт j**», где i в каждом шаге увеличивается на единицу, а j соответствует номеру пункта алгоритма.

Шаг 1, п. 1. В результате экспериментального моделирования работы КМУУ по заданной ГСА определяем вероятности выполнения каждой МК (табл. 1).

Таблица 1 – Вероятности выполнения микрокоманд ГСА G_1

a_i	$p(a_i)$	a_i	$p(a_i)$
a_1	0,0878	a_7	0,0923
a_2	0,1021	a_8	0,0923
a_3	0,1021	a_9	0,0408
a_4	0,0245	a_{10}	0,0408
a_5	0,3512	a_{11}	0,0088
a_6	0,0571		

Шаг 2, п. 2. Относим все ОЛЦ ко множеству нераспределенных ОЛЦ.

Шаг 3, п. 3. В качестве текущего выберем первый блок, начинающийся с адреса 0.

Шаг 4, п. 4. Текущий блок B_1 не является полностью заполненным.

Шаг 5, п. 5. Текущий блок B_1 полностью пустой. Определим «опасные» веса каждой нераспределенной ОЛЦ.

$v(O_1) = v(a_1) + v(a_2);$

$v(a_1) = p(a_5)\cdot(1–p(x_4))\cdot p(x_1)\cdot p(x_5) + p(a_8)\cdot p(x_1)\cdot p(x_5) + p(a_{10})\cdot(1–p(x_3))\cdot(1–p(x_2))\cdot p(x_5) =$
$= 0{,}3512\cdot(1–0{,}1)\cdot 0{,}2\cdot 0{,}9 + 0{,}0923\cdot 0{,}2\cdot 0{,}9 + 0{,}0408\cdot(1–0{,}5)\cdot(1–0{,}7)\cdot 0{,}9 =$
$= 0{,}0569 + 0{,}0166 + 0{,}0055 = 0{,}0790;$

$v(a_2) = p(a_{10})\cdot(1–p(x_3))\cdot p(x_2) = 0{,}0143;$

$v(O_1) = 0{,}0790 + 0{,}0143 = \mathbf{0{,}0933}.$

$v(O_2) = v(a_4) + v(a_5);$

$v(a_4) = p(a_3)\cdot(1–p(x_1))\cdot(1–p(x_2)) = 0{,}0245;$

$v(a_5) = p(a_8)\cdot(1–p(x_1)) = 0{,}0738;$

$v(O_2) = 0{,}0245 + 0{,}0738 = \mathbf{0{,}0983}.$

$v(O_3) = v(a_6) + v(a_7);$

$v(a_6) = p(a_3)\cdot(1–p(x_1))\cdot p(x_2) = 0{,}0572;$

$v(a_7) = p(a_5)\cdot p(x_4) = 0{,}1681\cdot 0{,}3 = 0{,}0351;$

$v(O_3) = 0{,}0572 + 0{,}0351 = \mathbf{0{,}0923}.$

$v(O_4) = v(a_9);$

$v(a_9) = p(a_3)\cdot p(x_1) = 0{,}0204;$

$v(O_4) = \mathbf{0{,}0204}.$

$v(O_5) = v(a_{11});$

$v(a_{11}) = p(a_5)\cdot(1–p(x_4))\cdot p(x_1)\cdot(1–p(x_5)) + p(a_8)\cdot p(x_1)\cdot(1–p(x_5))+$
$+ p(a_{10})\cdot(1–p(x_3))\cdot(1–p(x_2))\cdot(1–p(x_5)) = 0{,}0088;$

$v(O_5) = \mathbf{0{,}0088}.$

Поскольку на дальнейших шагах алгоритма полученные значения «опасных» весов ОЛЦ могут потребоваться снова, сведем их в таблицу (табл. 2).

Таблица 2 – «Опасные» веса ОЛЦ ГСА G_1

O_i	O_1	O_2	O_3	O_4	O_5
$v(O_i)$	0,0933	0,0983	0,0923	0,0204	0,0088

Согласно стратегии 3, выбираем ОЛЦ с максимальным «опасным» весом, т.е. O_2. Данную ОЛЦ добавляем в блок B_1 и размещаем, начиная с нулевого адреса (с ближайшего свободного адреса, допустимого для размещения ОЛЦ).

ОЛЦ O_2 считаем распределенной и исключаем из множества нераспределенных ОЛЦ. Поскольку пока O_2 – единственная в блоке, «опасный» вес блока $v(B_1)$ считаем равным $v(O_2)$.

Шаг 6, п. 6. Текущий блок B_1, способный вмещать две ОЛЦ, содержит только одну ОЛЦ O_2, то есть заполнен частично. Добавим в блок B_1 ту нераспределенную ОЛЦ, вместе с которой «опасный» вес блока будет наименьшим.

Согласно эвристике 3, добавление в блок B_1 еще одной ОЛЦ может снизить «опасные» веса каждой ОЛЦ, находящейся в блоке. Это возможно при условии, что существуют непосредственные микропрограммные переходы между ОЛЦ, находящимися в одном блоке.

Найдем «опасные» веса блока B_1 при добавлении к нему каждой из нераспределенных ОЛЦ.

1. Если добавляем ОЛЦ O_1.

Из O_1 в O_2 существуют следующие переходы:

– переход из a_3 в a_4: вес данного перехода равен

$$v(a_3 \rightarrow a_4) = p(a_3)\cdot(1-p(x_1))\cdot(1-p(x_2)) = 0{,}0245.$$

За счет «безопасности» этого перехода «опасный» вес O_2 снижается на величину $v(a_3 \rightarrow a_4)$ и становится равным $v(O_2) = 0{,}0983 - 0{,}0245 = 0{,}0738$.

Из O_2 в O_1 существуют следующие переходы:

– переход из a_5 в a_1: вес перехода равен

$$v(a_5 \rightarrow a_1) = p(a_5)\cdot(1-p(x_4))\cdot p(x_1)\cdot p(x_5) = 0{,}0569.$$

За счет «безопасности» этого перехода «опасный» вес O_1 снижается на величину $v(a_5 \rightarrow a_1)$ и становится равным $v(O_1) = 0{,}0933 - 0{,}0569 = 0{,}0364$.

При этом «опасный» вес блока B_1 будет равным

$$v(B_1) = v(O_2) + v(O_1) = 0{,}0738 + 0{,}0364 = \mathbf{0{,}1102}.$$

2. Если добавляем ОЛЦ O_3.

Из O_3 в O_2 существуют следующие переходы:

– переход из a_8 в a_5: вес перехода равен

$$v(a_8 \rightarrow a_5) = p(a_8)\cdot(1-p(x_1)) = 0{,}0738.$$

За счет «безопасности» этого перехода «опасный» вес O_2 снижается на величину $v(a_8 \rightarrow a_5)$ и становится равным $v(O_2) = 0{,}0983 - 0{,}0738 = 0{,}0245$.

Из O_2 в O_3 существуют следующие переходы:

– переход из a_5 в a_7: вес перехода равен

$$v(a_5 \rightarrow a_7) = p(a_5)\cdot p(x_4) = 0{,}0351.$$

За счет «безопасности» этого перехода «опасный» вес O_3 снижается на величину $v(a_5 \rightarrow a_6)$ и становится равным $v(O_3) = 0{,}0923 - 0{,}0351 = 0{,}0572$.

При этом «опасный» вес блока B_1 будет равным

$$v(B_1) = v(O_2) + v(O_3) = 0{,}0245 + 0{,}0572 = \mathbf{0{,}0817}.$$

3. Если добавляем ОЛЦ O_4.

Из O_2 в O_4 непосредственные переходы отсутствуют, следовательно, при размещении O_2 и O_4 в одном блоке «опасный» вес O_4 не уменьшается и остается равным $v(O_4) = 0{,}0204$.

Из O_4 в O_2 непосредственные переходы отсутствуют, следовательно, при размещении O_4 и O_2 в одном блоке «опасный» вес O_2 не уменьшается и остается равным $v(O_2) = 0{,}0983$.

При этом «опасный» вес блока B_1 будет равным

$v(B_1) = v(O_2) + v(O_4) = 0{,}0983 + 0{,}0204 =$ **0,1187**.

4. Если добавляем ОЛЦ O_5.

Из O_5 в O_2 непосредственные переходы отсутствуют, следовательно, при размещении O_5 и O_2 в одном блоке «опасный» вес O_2 не уменьшается и остается равным $v(O_2) = 0{,}0983$.

Из O_2 в O_5 существуют следующие переходы:

– переход из a_5 в a_{11}: вес перехода равен

$v(a_5 \rightarrow a_{11}) = p(a_5)\cdot(1-p(x_4))\cdot p(x_1)\cdot(1-p(x_5)) = 0{,}0063$.

За счет «безопасности» этого перехода «опасный» вес O_5 снижается на величину $v(a_5 \rightarrow a_1)$ и становится равным $v(O_5) = 0{,}0088 - 0{,}0063 = 0{,}0025$.

При этом «опасный» вес блока B_1 будет равным

$v(B_1) = v(O_2) + v(O_5) = 0{,}0983 + 0{,}0025 =$ **0,1008**.

Очевидно, что минимальное значение «опасного» веса блока будет получено при добавлении ОЛЦ O_3. Таким образом, цепь O_3 исключаем из множества нераспределенных ОЛЦ, добавляем в блок B_1 и размещаем в УП, начиная с адреса 4 (всегда кратно размеру блока УП).

Шаг 7, п. 7. Множество нераспределенных ОЛЦ не пусто. Переходим к п. 4.

Шаг 8, п. 4. Текущий блок B_1, способный вмещать две ОЛЦ, заполнен полностью. Переходим к блоку B_2 и считаем его текущим.

Шаг 9, п. 5. Блок B_2 пустой. Добавляем в него одну из нераспределенных ОЛЦ согласно стратегии 3. Согласно табл. 2, выбираем ОЛЦ O_1. Таким образом, ОЛЦ O_1 исключается из множества нераспределенных ОЛЦ, распределяется в блок B_2 и размещается в УП, начиная с адреса 8.

Шаг 10, п. 6. Текущий блок B_2, способный вмещать две ОЛЦ, содержит только одну ОЛЦ O_1, то есть заполнен частично. Добавим в блок B_2 ту нераспределенную ОЛЦ, вместе с которой «опасный» вес блока будет наименьшим.

Найдем «опасные» веса блока B_2 при добавлении к нему каждой из нераспределенных ОЛЦ.

1. Если добавляем ОЛЦ O_4.

Из O_1 в O_4 существуют следующие переходы:

– переход из a_3 в a_9: вес данного перехода равен

$v(a_3 \rightarrow a_9) = p(a_3)\cdot p(x_1) = 0{,}0204$.

За счет «безопасности» этого перехода «опасный» вес O_4 снижается на величину $v(a_3 \rightarrow a_9)$ и становится равным $v(O_4) = 0{,}0204 - 0{,}0204 = 0$.

Из O_4 в O_1 существуют следующие переходы:

– переход из a_{10} в a_2: вес перехода равен

$v(a_{10} \rightarrow a_2) = p(a_{10})\cdot(1-p(x_3))\cdot p(x_2) = 0{,}0143$.

– переход из a_{10} в a_1: вес перехода равен

$$v(a_{10} \rightarrow a_1) = p(a_{10}) \cdot (1-p(x_3)) \cdot (1-p(x_2)) \cdot p(x_5) = 0{,}0055.$$

За счет «безопасности» этого перехода «опасный» вес O_1 снижается на величину $v(a_{10} \rightarrow a_2)$ и становится равным $v(O_1) = 0{,}0933 - 0{,}0143 - 0{,}0055 = 0{,}0735$.

При этом «опасный» вес блока B_2 будет равным

$$v(B_2) = v(O_4) + v(O_1) = 0 + 0{,}0735 = \mathbf{0{,}0735}.$$

2. *Если добавляем ОЛЦ O_5.*

Из O_1 в O_5 непосредственные переходы отсутствуют, следовательно, при размещении O_1 и O_5 в одном блоке «опасный» вес O_5 не уменьшается и остается равным $v(O_5) = 0{,}0088$.

Из O_5 в O_1 непосредственные переходы отсутствуют, следовательно, при размещении O_5 и O_1 в одном блоке «опасный» вес O_1 не уменьшается и остается равным $v(O_1) = 0{,}0933$.

При этом «опасный» вес блока B_2 будет равным

$$v(B_2) = v(O_1) + v(O_5) = 0{,}0933 + 0{,}0088 = \mathbf{0{,}1021}.$$

Очевидно, что минимальное значение «опасного» веса блока будет получено при добавлении ОЛЦ O_4. Таким образом, цепь O_4 исключаем из множества нераспределенных ОЛЦ, добавляем в блок B_2 и размещаем в УП, начиная с адреса 12.

Шаг 11, п. 7. Множество нераспределенных ОЛЦ не пусто. Переходим к пункту 4.

Шаг 12, п. 4. Текущий блок B_2, способный вмещать две ОЛЦ, заполнен полностью. Переходим к блоку B_3 и считаем его текущим.

Шаг 13, п. 5. Блок B_3 пустой. Добавляем в него одну из нераспределенных ОЛЦ согласно стратегии 3. Единственной нераспределенной ОЛЦ является O_5, которая исключается из множества нераспределенных ОЛЦ, добавляется в блок B_3, начиная с адреса 16.

Шаг 14, п. 6. Блок B_3 заполнен частично, однако множество нераспределенных ОЛЦ пусто, поэтому ничего не добавляем.

Шаг 15, п. 7. Множество нераспределенных ОЦЛ пусто, переходим к п. 8.

Шаг 16, п. 8. Результатом работы алгоритма стало следующее размещение ОЛЦ в адресном пространстве УП (табл. 3).

Таблица 3 – Адресация ОЛЦ после оптимизации (G_1)

Адрес	ОЛЦ	Адрес	ОЛЦ
0	O_2	12	O_4
4	O_3	16	O_5
8	O_1		

Отметим, что согласно результатам моделирования, при кэш-памяти размером 1 строка на 8 слов полученное размещение обеспечивает вероятность кэш-попаданий $p_h = 0{,}8273$. При последовательном (неоптимизированном) размещении ОЛЦ экспериментальное значение $p_h = 0{,}7596$. Таким образом, использование предложенного эвристического алгоритма размещения ОЛЦ в управляющей памяти привело к повышению эффективности использования модуля кэш-памяти почти на 9 %.

Заключение

Разработанный эвристический алгоритм оптимизации размещения микрокоманд в управляющей памяти позволяет повысить эффективность использования модуля кэш-памяти в структуре КМУУ с разделением кодов, что положительно отражается на быстродействии логической схемы устройства. Практическая реализация данного алгоритма возможна в специализированных САПР цифровых устройств управления.

В перспективе дальнейшей научной работы авторы видят исследование особенностей применения алгоритма для различных структур КМУУ с разделением кодов, имеющих разные архитектурные типы модулей кэш-памяти.

Литература

1. Баркалов А.А., Палагин А.В. Синтез микропрограммных устройств управления. – Киев: Институт кибернетики НАН Украины, 1997. – 135 с.
2. Баркалов А.А. Синтез устройств управления на программируемых логических устройствах. – Донецк: ДонНТУ, 2002. – 262 с.
3. Баркалов А.А., Ковалев С.А., Бабаков Р.М., Николаенко Д.В. Организация композиционных микропрограммных устройств управления с разделением кодов и кэш-памятью // Искусственный интеллект. – 2007. – № 3. – С. 135-138.

О.О. Баркалов, С.О. Ковальов, Р.М. Бабаков, Д.В. Ніколаєнко

Евристичний алгоритм оптимізації розміщення мікрокоманд в композиційному мікропрограмному пристрої керування із розподілом кодів та кеш-пам'яттю

Розроблено евристичний алгоритм збільшення ефективності використання модуля кеш-пам'яті у композиційному мікропрограмному пристрої керування із розподілом кодів, заснований на спеціальній адресації операторних лінійних кіл. Запропонований ряд стратегій поєднання кількох операторних кіл в одному блоці пам'яті, що дозволяє у загальному випадку збільшити значення імовірності кеш-попадань для граф-схеми реалізованого алгоритму керування. Розглянутий приклад використання запропонованого евристичного алгоритму.

A.A. Barkalov, S.A. Kovalev, R.M. Babakov, D.V. Nikolaenko

The Heuristic Algorithm of Optimization of Placement of Microinstructions in Compositional Microprogram Control Unit with Division of Codes and Cache-memory

The heuristic algorithm for increased efficiency of cache-memory module usage in compositional microprogram control unit, based on special addressing of operator linear chains, is developed. The number of strategies to combine some operator linear chains in one memory block are proposed; they allow in common case to increase value of probability of cache hits for given flow-chart. The example of using of proposed heuristic algorithm is given.

Статья поступила в редакцию 03.12.2007.